The information management and database system of the present invention comprises a flexible, self-referential table that stores data. The table of the present invention may store any type of data, both structured and unstructured, and provides an interface to other application programs. The table of the present invention comprises a plurality of rows and columns. Each row has an object identification number (OID) and each column also has an OID. A row corresponds to a record and a column corresponds to a field such that the intersection of a row and a column comprises a cell that may contain data for a particular record related to a particular field, a cell may also point to another record. To enhance searching and to provide for synchronization between columns, columns are entered as rows in the table and the record corresponding to a column contains various information about the column. The table includes an index structure for extended queries.

US PAT NO: 4,979,098 [IMAGE AVAILABLE] L12: 3 of 3
TITLE: Multiple address space token designation, protection controls, designation translation and lookaside

ABSTRACT:
A method and apparatus is provided to translate the contents of access registers into information for use in performing addressing functions for multiple virtual address spaces. The access registers represent the full addressing capability of the system but do not directly contain the addressing information. The system has a plurality of general purpose registers, a plurality of access registers associated with the general registers, an access list having access list entries which is addressed by the contents of the access register, memory storage for holding address space number second table entries (ASTE), where the contents of the access list entry locate the ASTE and where the ASTE contains the addressing information needed to translate a virtual address when combined with the contents of a general purpose register. Access register translation (ART) consists of the process of determining addressing information by using the access list entry and the ASTE. The results of the ART process are stored in an ART lookaside buffer (ALB) which stores the results of ART while valid for later use.

=> d his

(FILE 'USPAT' ENTERED AT 12:42:32 ON 18 SEP 1998)
L1 10 S (5,564,046 OR 5,557,787 OR 5,553,218 OR 5,537,633 OR 5,537,
L2 443 S 707/1/CCLS
L3 420 S 707/3/CCLS
L4 279 S 707/4/CCLS
L5 249 S 707/102/CCLS
L6 1224 S L2 OR L3 OR L4 OR L5
L7 54 S LOGICAL# TABLE AND (ROW? OR COLUMN? OR TUPLE? ATTRIBUTE# OR
L8 54 S LOGICAL# TABLE AND (ROW? OR COLUMN? OR TUPLE? OR ATTRIBUTE#
L9 61 S LOGICAL# TABLE? AND (ROW? OR COLUMN? OR TUPLE? OR ATTRIBUTE
L10 0 S (LOGICAL# TABLE? AND (ROW? OR COLUMN? OR TUPLE? OR ATTRIBUT
L11 0 S (LOGICAL# TABLE? AND (ROW? OR COLUMN? OR TUPLE? OR ATTRIBUT
L12 3 S (LOGICAL# TABLE? AND (ROW? OR COLUMN? OR TUPLE? OR ATTRIBUT

=> s l6 and l9

L13 7 L6 AND L9

=> d l13 1-7 ti ab

US PAT NO: 5,809,500 [IMAGE AVAILABLE] L13: 1 of 7
TITLE: System for converting programs and databases to correct year 2000 processing errors

ABSTRACT:
A method for processing or modifying programs and databases containing abbreviated date **fields** to achieve year 2000 compliancy includes examining an original database to ascertain the location of date **fields**, creating a supplementary PALM file that includes fully expanded date **fields** representative of corresponding date **fields** in the original database and modifying instructions or modules of the original program to access the PALM file for date information instead of the original database. The PALM file contains fully expanded date representations of abbreviated date **fields** of the original database. The PALM file also includes record **identifiers** and **field identifiers** to enable the modified program to access the required date information. Portions of the program requiring modification are identified by scanning for keywords that are generated from an examination of the database and/or from **field** layout definitions. The program is also examined for related keyword constructs or redefinitions found in the program itself. When converted, the modified program accesses the PALM file for date processing and accesses the original database for other processing steps. The method enables a computational system to determine the periods between dates in a non-ambiguous manner and yet advantageously uses the original database to retain system compatibility.

US PAT NO: 5,794,229 [IMAGE AVAILABLE] L13: 2 of 7
TITLE: Database system with methodology for storing a database table by vertically partitioning all **columns** of the table

ABSTRACT:
A Client/Server Database System with improved methods for performing database queries, particularly DSS-type queries, is described. The system includes one or more Clients (e.g., Terminals or PCs) connected via a Network to a Server. In general operation, Clients store data in and retrieve data from one or more database tables resident on the Server by submitting SQL commands, some of which specify "queries"--criteria for selecting particular records of a table. The system implements methods for storing data vertically (i.e., by **column**), instead of horizontally (i.e., by **row**) as is traditionally done. Each **column** comprises a plurality of "cells" (i.e., **column** value for a record), which are arranged on a data page in a contiguous fashion. By storing data in a **column**-wise basis, the system can process a DSS query by bringing in only those **columns** of data which are of interest. Instead of retrieving **row**-based data pages consisting of information which is largely not of interest to a query, **column**-based pages can be retrieved consisting of information which is mostly, if not completely, of interest to the query. The retrieval itself can be done using more-efficient large block I/O transfers. The system includes data compression which is provided at the level of Cache or Buffer Managers, thus providing on-the-fly data compression in a manner which is transparent to each object. Since vertical storage of data leads to high repetition on a given data page, the system provides improved compression/decompression.

US PAT NO: 5,781,905 [IMAGE AVAILABLE] L13: 3 of 7

TITLE: Program generating method combining data item part with database manipulation part

ABSTRACT:
Data item parts and database manipulation parts are prepared in advance, and a business transaction program is generated by combining the data item parts and the database manipulation parts in accordance with a business transaction specification. The data item part includes a data unit and a procedure unit, which is provided for each data item to be processed by the business transaction program, the data unit storing data to be processed, and the procedure unit storing procedure information defining the process contents of the data to be processed. The database manipulation part includes a data unit and a procedure unit, which is provided for each table of a database for storing data to be processed by the business transaction program, the procedure unit storing procedure information defining database manipulation, the database manipulation including addition, update, deletion, search, or the like of a database record, the data unit storing data of the database record and a pointer to the data item part for processing the data in the database record.

US PAT NO: 5,734,887 [IMAGE AVAILABLE] L13: 4 of 7
TITLE: Method and apparatus for logical data access to a physical relational database

ABSTRACT:
Logical Data Access to the physical structure of a relational database is provided for one or more Applications. Applications are developed using the logical entity types and logical entity type **attribute** names as described in a logical data model. The Applications then use a Logical Data Access Interface to access each of the required physical relational database tables via the Logical Data Access Layer. Applications then use logical entity type and logical entity type **attribute** names as specified in the Logical Data Model in making Logical Data Requests to the Logical Data Access Layer. The Logical Data Access Layer provides a rich set of functions for allowing an Application to control and manage a database, build and execute database queries and interface with physical database. The Logical Data Access Layer determines which of the physical tables and associated **columns** are required to satisfy the Application request and then builds one or more database query statements containing the appropriate physical table and **column** names.

US PAT NO: 5,729,730 [IMAGE AVAILABLE] L13: 5 of 7
TITLE: Method and apparatus for improved information storage and retrieval system

ABSTRACT:
The information management and database system of the present invention comprises a flexible, self-referential table that stores data. The table of the present invention may store any type of data, both structured and unstructured, and provides an interface to other application programs. The table of the present invention comprises a plurality of **rows** and **columns**. Each **row** has an object **identification** number (OID) and each **column** also has an OID. A **row** corresponds to a record and a **column** corresponds to a **field** such that the intersection of a **row** and a **column** comprises a cell that may contain data for a particular record related to a particular **field**, a cell may also point to another record. To enhance searching and to provide for synchronization between **columns**, **columns** are entered as **rows** in the table and the record corresponding to a **column** contains various information about the **column**. The table includes an **index** structure for extended queries.

US PAT NO: 5,701,453 [IMAGE AVAILABLE] L13: 6 of 7

TITLE: Logical schema to allow access to a relational database without using knowledge of the database structure

ABSTRACT:
Logical schemas are used to allow an end user the ability to access and manipulate relational database data without knowledge of the structure of the relational database. A logical schema is first created specifying which tables are available to an end user, and the relationships between **columns** of those tables. The logical schema defines a structure for the data **fields** having a master level and a plurality of detail levels. An end user may manipulate the logical schema using a graphical interface to build customized forms, reports, and queries. An end user is not required to be familiar with a database query language, such as SQL, or the structure of the relational database.

US PAT NO: 5,418,950 [IMAGE AVAILABLE] L13: 7 of 7
TITLE: System for interactive clause window construction of SQL queries

ABSTRACT:
A method and system for viewing information stored in one or more **rows** and **columns** in a database. The system having a logical progression of choosing a **column**, determining conditions for a **row** to be included in a panel, specifying a **column** to be used as a base to group **rows** in a panel and a display for viewing the information.

```
=> s 707/1/ccls

L2          443 707/1/CCLS

=>

=> s 707/3/ccls

L3          420 707/3/CCLS

=> s 707/4/ccls

L4          279 707/4/CCLS

=> s 707/102/ccls

L5          249 707/102/CCLS

=> s l2 or l3 or l4 or l5

L6         1224 L2 OR L3 OR L4 OR L5

=> s logical# table and (row? or column? or tuple? attribute# or field?) and
(id or identification# or identifier# or index?)

          61530 LOGICAL#
         543197 TABLE
            101 LOGICAL# TABLE
                  (LOGICAL#(W)TABLE)
         174306 ROW?
         287556 COLUMN?
           1147 TUPLE?
          70829 ATTRIBUTE#
             12 TUPLE? ATTRIBUTE#
                  (TUPLE?(W)ATTRIBUTE#)
        1184475 FIELD?
          44753 ID
         102328 IDENTIFICATION#
          16471 IDENTIFIER#
         185262 INDEX?
L7           54 LOGICAL# TABLE AND (ROW? OR COLUMN? OR TUPLE? ATTRIBUTE# OR
 FI
                ELD?) AND (ID OR IDENTIFICATION# OR IDENTIFIER# OR INDEX?)

=> s logical# table and (row? or column? or tuple? or attribute# or field?)
and (id or identification# or identifier# or index?)

          61530 LOGICAL#
         543197 TABLE
            101 LOGICAL# TABLE
                  (LOGICAL#(W)TABLE)
         174306 ROW?
         287556 COLUMN?
           1147 TUPLE?
          70829 ATTRIBUTE#
```

```
       1184475 FIELD?
         44753 ID
        102328 IDENTIFICATION#
         16471 IDENTIFIER#
        185262 INDEX?
L8          54 LOGICAL# TABLE AND (ROW? OR COLUMN? OR TUPLE? OR ATTRIBUTE#
 OR
                FIELD?) AND (ID OR IDENTIFICATION# OR IDENTIFIER# OR INDEX
?)

=> s logical# table? and (row? or column? or tuple? or attribute# or field?)
and (id or identification# or identifier# or index?)

         61530 LOGICAL#
        580358 TABLE?
           112 LOGICAL# TABLE?
                 (LOGICAL#(W)TABLE?)
        174306 ROW?
        287556 COLUMN?
          1147 TUPLE?
         70829 ATTRIBUTE#
       1184475 FIELD?
         44753 ID
        102328 IDENTIFICATION#
         16471 IDENTIFIER#
        185262 INDEX?
L9          61 LOGICAL# TABLE? AND (ROW? OR COLUMN? OR TUPLE? OR ATTRIBUTE
# O
              R FIELD?) AND (ID OR IDENTIFICATION# OR IDENTIFIER# OR INDE
X?)

=> s (logical# table? and (row? or column? or tuple? or attribute# or field?)
and (id or identification# or identifier# or index?))/ti

           423 LOGICAL#/TI
          7622 TABLE?/TI
             0 LOGICAL# TABLE?/TI
                 ((LOGICAL#(W)TABLE?)/TI)
          1254 ROW?/TI
          2787 COLUMN?/TI
             6 TUPLE?/TI
           172 ATTRIBUTE#/TI
          9349 FIELD?/TI
           167 ID/TI
          3022 IDENTIFICATION#/TI
           229 IDENTIFIER#/TI
          2770 INDEX?/TI
L10          0 (LOGICAL# TABLE? AND (ROW? OR COLUMN? OR TUPLE? OR ATTRIBUT
E#
              OR FIELD?) AND (ID OR IDENTIFICATION# OR IDENTIFIER# OR IND
EX?
              ))/TI

=> s (logical# table? and (row? or column? or tuple? or attribute# or field?)
and (id or identification# or identifier# or index?))/ab

          4032 LOGICAL#/AB
         27159 TABLE?/AB
             1 LOGICAL# TABLE?/AB
                 ((LOGICAL#(W)TABLE?)/AB)
         23774 ROW?/AB
         24092 COLUMN?/AB
            82 TUPLE?/AB
```

```
          2122 ATTRIBUTE#/AB
         60271 FI[redacted]?/AB
          1637 ID/AB
          9553 IDENTIFICATION#/AB
          1676 IDENTIFIER#/AB
         19603 INDEX?/AB
L11           0 (LOGICAL# TABLE? AND (ROW? OR COLUMN? OR TUPLE? OR ATTRIBUT
E#
                OR FIELD?) AND (ID OR IDENTIFICATION# OR IDENTIFIER# OR IND
EX?
                ))/AB

=> s (logical# table? and (row? or column? or tuple? or attribute# or field?)
and (id or identification# or identifier# or index?))/clm

         10890 LOGICAL#/CLM
         50712 TABLE?/CLM
             7 LOGICAL# TABLE?/CLM
                 ((LOGICAL#(W)TABLE?)/CLM)
         53912 ROW?/CLM
         46933 COLUMN?/CLM
           184 TUPLE?/CLM
          3355 ATTRIBUTE#/CLM
         92875 FIELD?/CLM
          7330 ID/CLM
         18319 IDENTIFICATION#/CLM
          4623 IDENTIFIER#/CLM
         42209 INDEX?/CLM
L12           3 (LOGICAL# TABLE? AND (ROW? OR COLUMN? OR TUPLE? OR ATTRIBUT
E#
                OR FIELD?) AND (ID OR IDENTIFICATION# OR IDENTIFIER# OR IND
EX?
                ))/CLM

=> d l12 1-3 ti ab


US PAT NO:      5,734,887 [IMAGE AVAILABLE]             L12: 1 of 3
TITLE:          Method and apparatus for logical data access to a physical
                  relational database

ABSTRACT:
Logical Data Access to the physical structure of a relational database is
provided for one or more Applications. Applications are developed using
the logical entity types and logical entity type attribute names as
described in a logical data model. The Applications then use a Logical
Data Access Interface to access each of the required physical relational
database tables via the Logical Data Access Layer. Applications then use
logical entity type and logical entity type attribute names as specified
in the Logical Data Model in making Logical Data Requests to the Logical
Data Access Layer. The Logical Data Access Layer provides a rich set of
functions for allowing an Application to control and manage a database,
build and execute database queries and interface with physical database.
The Logical Data Access Layer determines which of the physical tables and
associated columns are required to satisfy the Application request and
then builds one or more database query statements containing the
appropriate physical table and column names.

US PAT NO:      5,729,730 [IMAGE AVAILABLE]             L12: 2 of 3
TITLE:          Method and apparatus for improved information storage and
                  retrieval system

ABSTRACT:
```